Soluções para o Problema da Mochila Binária: Uma Comparação entre Estratégia Gulosa e Transformar para Conquistar

Gabriel Lenser[1], Luiz E. O. de Oliveira[1], Pedro H. O. Berti[1]

[1]Colegiado de Ciência da Computação
Campus de Cascavel - UNIOESTE

Resumo. *Este artigo realiza uma análise comparativa entre duas estratégias para resolver o problema da mochila binária: a abordagem gulosa clássica e a estratégia transformar para conquistar com ordenação inicial dos itens. Foram implementados e avaliados dois algoritmos em linguagem C, com testes empíricos sobre quinze instâncias de diferentes tamanhos. A estratégia gulosa apresentou complexidade $O(n^2)$ e tempos de execução crescentes conforme o número de itens, com oscilações influenciadas pela distribuição dos dados. Já a estratégia transformar para conquistar, com complexidade $O(n \log n)$, demonstrou desempenho significativamente superior, mantendo tempos muito baixos e estáveis mesmo em instâncias de grande porte.*

1. Introdução

O Algoritmo Guloso é um algoritmo comumente utilizado para fornecer soluções à problemas práticos. Esse algoritmo foi proposto em 1959 pelo cientista Wyb Dijsr para solucionar problemas relacionados a grafos. Uma das suas principais características é que a cada etapa, ele toma a solução ótima local no estado atual [Wang 2023]. Esse algoritmo é considerado um dos algoritmos mais utilizados, ademais ele utiliza menos poder computacional e quantidade de algorítmos em comparação com outras heurísticas [Durmus et al. 2019].

O problema da mochila binária é um típico NP completo. Esse problema pode ser resolvido de diferentes formas como: recursividade utilizando *backtracking*, *branch and bound*, programação dinâmica e algoritmo guloso [Abidin 2017]. Contudo, a mochila binária é um problema de otimização combinatória, onde temos que maximizar o benefício total em uma mochila, sem exceder a sua capacidade. Dentro da mochila, cada item é categorizado pelo seu peso e seu valor de lucro e a mochila é caracterizada pela sua capacidade. O caso ótimo desse algoritmo é quando o peso total é menor ou igual a capacidade da mochila e se obtém o maior valor possível de benefício [Pradhan et al. 2014]

O algoritmo guloso para a mochila consiste em três passos: calcular a razão entre o lucro e o peso de cada item, para se obter o valor unitário; ordenar os itens de forma decrescente, de acordo com seus respectivos valores unitários; e por último, selecionar os itens com maior valor unitário até a mochila não poder carregar um item novo ou quando todos os itens já foram inseridos na mochila [Wang 2023].

2. Objetivo

O objetivo deste trabalho é comparar e analisar duas abordagens diferentes para o problema da mochila binária, com a utilização da estratégia gulosa, considerando fatores como os polinômios de custo assintótico, o tempo cronológico de execução e as soluções obtidas. O foco é comparar uma implementação que pré-ordena os valores unitários dos itens com uma que não realiza essa pré-ordenação, em diferentes conjuntos de dados.

3. Metodologia

A avaliação comparativa das duas implementações foi conduzida utilizando a linguagem C, escolhida por suas características como linguagem compilada que proporciona maior eficiência na execução. Os experimentos foram realizados em uma máquina com processador AMD Ryzen 7 7435HS 3.10 GHz, 16 GB de RAM, sistema operacional Windows 11, utilizando o compilador GCC sem flags de otimização -O2. Os testes focaram na análise da relação entre o tamanho da instância do problema da mochila e o tempo de execução correspondente. Para garantir a confiabilidade dos resultados, cada configuração foi executada seis vezes, sendo a primeira execução descartada. A média aritmética foi então calculada a partir das cinco execuções restantes. Esta metodologia permite uma comparação objetiva do desempenho computacional das implementações propostas.

3.1. Análise Assintótica e Empírica do Algoritmo

A análise de complexidade assintótica das implementações foi conduzida através de metodologias distintas, adequadas às características específicas de cada função. Para as operações de medição de tempo de execução, estas não foram contabilizadas no cálculo da complexidade assintótica, uma vez que representam instrumentação para coleta de métricas e não fazem parte da lógica algorítmica principal.

Para o algoritmo guloso sem pré-ordenação (modo 1), a análise foi realizada através da contagem direta do número de operações, devido ao fato de ser uma função iterativa. O algoritmo apresenta uma estrutura de *loops* aninhados onde o *loop* externo executa no máximo n iterações (uma para cada item) e o *loop* interno percorre todos os n itens em cada iteração, buscando o item com maior custo-benefício ainda não processado. A contagem detalhada das operações elementares resultou no polinômio $14n^2 + 20n + 6$, caracterizando complexidade assintótica $O(n^2)$ no pior caso.

Para o algoritmo *Quick Sor*t utilizado na estratégia transformar para conquistar (modo 2), foi empregada a análise através de equação de recorrência. A relação de recorrência estabelecida é $T(n) = 2T(n/2) + 15n + 37$ para n maior que 1, com caso base $T(1) = 1$. Aplicando o Teorema Mestre, onde $a = 2$, $b = 2$ e $f(n) = 15n + 37$, temos que n elevado ao logaritmo de 2 na base 2 resulta em n. Como $f(n) = O(n)$, enquadra-se no caso 2 do teorema, resultando em complexidade $O(n \log n)$. A estratégia transformar para conquistar combina a ordenação $O(n \log n)$ com a seleção sequencial dos itens ordenados, que apresenta complexidade $O(n)$, resultando em complexidade $O(n \log n)$ dominada pela fase de ordenação.

3.2. Implementação

A implementação desenvolvida aborda o problema clássico da mochila binária através de duas estratégias algorítmicas distintas: o algoritmo guloso e a estratégia transformar para conquistar. O programa foi estruturado em linguagem C, utilizando uma abordagem modular que permite a comparação direta entre as diferentes metodologias de resolução.

A estrutura de dados central da implementação é representada pela struct Item, que encapsula as propriedades essenciais de cada elemento: valor, peso, custo-benefício, status de seleção e status de descarte. Esta modelagem permite um controle granular do estado de cada item durante a execução dos algoritmos, facilitando tanto a implementação quanto a análise dos resultados.

O algoritmo guloso implementado segue uma estratégia de seleção iterativa baseada na maximização da relação custo-benefício. Em cada iteração, o algoritmo percorre todos os itens disponíveis, identificando aquele com a maior razão valor/peso que ainda não foi processado. A decisão de inclusão é tomada verificando se a adição do item não excede a capacidade da mochila. Caso o item possa ser incluído, ele é marcado como selecionado e seu peso é adicionado ao peso total, caso contrário, é marcado como descartado. Este processo continua até que não existam mais itens possíveis para seleção ou a capacidade da mochila seja atingida.

A estratégia transformar para conquistar emprega uma abordagem diferenciada, onde o problema original é modificado em uma forma mais fácil de resolver e então solucionado [Brun 2025]. Especificamente, inicia-se com uma fase de pré-processamento onde todos os itens são ordenados em ordem decrescente de relação custo-benefício utilizando o algoritmo *Quick Sort*. O *Quick Sort* é um algoritmo de ordenação baseado na estratégia dividir para conquistar, que consiste em fragmentar o problema em subproblemas menores, resolvê-los recursivamente e combinar as soluções parciais para obter a solução final [Folador et al. 2014] . Especificamente, o algoritmo funciona selecionando um elemento como pivô e reorganizando o *array* de forma que elementos maiores que o pivô fiquem à sua esquerda e elementos menores à sua direita. O processo é aplicado recursivamente às sublistas resultantes até que todo o conjunto esteja ordenado. A implementação do *Quick Sort* incorpora uma otimização através da seleção aleatória do pivô, reduzindo a probabilidade de casos degenerados que resultariam em complexidade $O(n^2)$. Após a ordenação, o algoritmo percorre sequencialmente os itens ordenados, selecionando cada um que couber na capacidade restante da mochila.

Para avaliação de performance, o programa implementa medição precisa do tempo de execução utilizando a biblioteca *time*. O cronômetro é iniciado antes da execução do algoritmo principal e finalizado após sua conclusão, fornecendo métricas em segundos com precisão de nanosegundos. Esta instrumentação permite análises comparativas objetivas entre as diferentes estratégias implementadas.

A estruturação do código permite a execução seletiva dos algoritmos através de parâmetros de linha de comando. O programa processa automaticamente arquivos de

entrada de diferentes tamanhos, possibilitando estudos escalares da performance algorítmica em função do número de itens e da capacidade da mochila.

A estruturação do código permite a execução seletiva dos algoritmos através de parâmetros de linha de comando. O programa processa automaticamente arquivos de entrada de diferentes tamanhos, possibilitando estudos escalares da performance algorítmica em função do número de itens e da capacidade da mochila. A execução do programa compilado segue a sintaxe: .\Binary-Knapsack-Greedy-TC.exe [modo] [total_itens] [arquivo_entrada], onde o parâmetro [modo] aceita valor 1 para execução do algoritmo guloso sem pré-ordenação ou valor 2 para a estratégia transformar para conquistar, [total_itens] especifica a quantidade de itens na instância do problema, e [arquivo_entrada] indica o caminho para o arquivo de dados de teste. O código fonte completo, incluindo os detalhes dos cálculos dos polinômios de complexidade e a lógica de implementação comentada, está disponível no repositório: https://github.com/EduardoGarzon/Binary-Knapsack-Study.

No repositório, podem ser consultados os detalhes dos cálculos dos polinômios de complexidade, incluindo a contagem das operações, bem como a lógica completa de implementação dos algoritmos com comentários sobre cada etapa do processo de análise assintótica.

3.3. Testes

Para avaliar o comportamento das abordagens implementadas, foi utilizado um conjunto abrangente de instâncias de teste que varia sistematicamente o tamanho do problema da mochila binária. As instâncias foram estruturadas seguindo um formato padronizado, onde a primeira linha do arquivo contém a capacidade máxima da mochila, seguida por uma linha com os valores (benefícios) de todos os itens e, posteriormente, uma linha com os respectivos pesos (custos) dos itens.

O conjunto de testes compreende quinze instâncias distintas, denominadas Mochila10, Mochila50, Mochila100, Mochila200, Mochila300, Mochila500, Mochila750, Mochila1000, Mochila1250, Mochila1500, Mochila2000, Mochila2500, Mochila3000, Mochila4000 e Mochila5000, onde a numeração indica o número de itens disponíveis para seleção em cada instância. Esta distribuição foi estruturada para permitir uma análise escalável do desempenho algorítmico, iniciando com problemas de pequena dimensão (10 itens) e estendendo-se até instâncias de grande porte (5.000 itens).

A distribuição dos tamanhos das instâncias segue uma progressão que intensifica gradualmente a dimensão do problema, abrangendo desde cenários de aplicação básica até problemas de dimensões significativas. As instâncias menores (10 a 100 itens) representam problemas de escala reduzida, enquanto as instâncias intermediárias (200 a 1.000 itens) configuram cenários de escala moderada. As instâncias de maior dimensão (1.250 a 5.000 itens) constituem problemas de grande escala, adequados para avaliar diferenças de desempenho entre os algoritmos e identificar padrões de crescimento temporal.

Esta metodologia de teste escalonada permite uma avaliação sistemática da complexidade empírica dos algoritmos, possibilitando a identificação de padrões de comportamento computacional e a verificação empírica das complexidades teóricas esperadas para cada estratégia algorítmica implementada. Esta variedade de escalas possibilita uma análise abrangente do comportamento algorítmico em diferentes contextos de aplicação.

4. Resultados e Discussão

Esta seção apresenta e analisa os resultados obtidos para as duas abordagens implementadas: a estratégia gulosa e a abordagem transformar para conquistar (TC). Inicialmente, são exibidos os tempos de execução registrados para cada instância em múltiplas repetições, seguidos das médias calculadas para reduzir o impacto de variações pontuais. Em seguida, realiza-se uma comparação direta entre as estratégias, destacando padrões observados, variações inesperadas e a relação com as complexidades teóricas esperadas. Por fim, gráficos são apresentados para ilustrar visualmente as tendências de desempenho em função do tamanho das instâncias.

O Quadro 1 mostra os tempos de execução da estratégia gulosa nos Testes 1 a 3. Como esperado, o tempo aumenta conforme cresce a quantidade de itens na mochila, o que está de acordo com a complexidade $O(n^2)$ desse algoritmo. Em instâncias pequenas, como com 10, 50 ou 100 itens, os tempos foram praticamente iguais entre os testes, o que indica um comportamento bem estável. A partir de instâncias maiores, os tempos variam um pouco entre as repetições, o que pode acontecer por conta do funcionamento interno do computador, como o uso de memória ou outros programas rodando ao mesmo tempo. O Teste 1 foi descartado, seguindo o que foi definido na metodologia, já que a primeira execução pode ter pequenas diferenças por conta da preparação inicial do programa. No geral, os resultados mostram que o tempo do algoritmo cresce de forma rápida conforme o problema aumenta, como era esperado, e mostram que os testes foram realizados de forma consistente.

INSTÂNCIA	TESTE 1 (DESCARTADO)	TESTE 2	TESTE 3
10	0,00000120s	0,00000160s	0,00000160s
50	0,00001150s	0,00001200s	0,00001190s
100	0,00002460s	0,00002460s	0,00002460s
200	0,00011660s	0,00011900s	0,00011850s
300	0,00024310s	0,00024590s	0,00024378s
500	0,00106080s	0,00057580s	0,00067570s
750	0,00200080s	0,00133668s	0,00154960s
1000	0,00139360s	0,00139390s	0,00139380s
1250	0,00417600s	0,00438150s	0,00367870s
1500	0,00124510s	0,00126540s	0,00124660s
2000	0,00843630s	0,00792820s	0,00764860s
2500	0,00490670s	0,00621090s	0,00466590s
3000	0,00533720s	0,00575680s	0,00599818s
4000	0,02660578s	0,02653570s	0,02525210s
5000	0,02970800s	0,03168310s	0,02879790s

Quadro 1. Testes 1 ao 3 utilizando estratégia gulosa

O Quadro 2 apresenta os resultados dos Testes 4 ao 6 utilizando a estratégia gulosa. Assim como observado nos testes anteriores, os tempos de execução aumentam conforme o número de itens na mochila cresce, o que reforça o comportamento esperado para algoritmos com complexidade quadrática. Em instâncias pequenas, como 10, 50 e 100 itens, os tempos são bastante próximos entre si, mostrando estabilidade. Já em instâncias maiores, aparecem variações mais visíveis entre os testes. Por exemplo, na instância com 750 itens, o tempo variou de 0,00136s a 0,00185s.

De forma geral, os testes confirmam que a estratégia gulosa apresenta um crescimento acelerado de tempo à medida que o problema se torna maior, e reforçam a importância de realizar múltiplas medições para obter resultados mais confiáveis.

INSTÂNCIA	TESTE 4	TESTE 5	TESTE 6
10	0,00000150s	0,00000170s	0,00000150s
50	0,00001220s	0,00001210s	0,00001170s
100	0,00003090s	0,00002450s	0,00003650s
200	0,00011760s	0,00011800s	0,00011940s
300	0,00032780s	0,00024510s	0,00024470s
500	0,00064300s	0,00055940s	0,00055420s
750	0,00139880s	0,00185610s	0,00136840s
1000	0,00141840s	0,00141750s	0,00149000s
1250	0,00398550s	0,00389820s	0,00374100s
1500	0,00126430s	0,00126610s	0,00124478s
2000	0,00833890s	0,00817120s	0,00761430s
2500	0,00477360s	0,00465888s	0,00438790s
3000	0,00527940s	0,00527278s	0,00577670s
4000	0,02521140s	0,02383740s	0,02755118s
5000	0,02828740s	0,03331300s	0,02757070s

Quadro 2. Testes 4 ao 6 utilizando estratégia gulosa

O Quadro 3 mostra a média dos tempos de execução dos Testes 2 ao 6 para cada instância utilizando a estratégia gulosa. Essa média foi calculada justamente para representar o comportamento mais estável do algoritmo, desconsiderando possíveis variações do primeiro teste. Observa-se que os tempos médios aumentam de forma progressiva conforme o número de itens na instância cresce, o que é compatível com o comportamento teórico do algoritmo, que tem complexidade $O(n^2)$.

As instâncias menores, como 10 e 50 itens, apresentam tempos praticamente desprezíveis (na casa dos microssegundos). A partir de 500 itens, já é possível notar um crescimento mais expressivo, chegando a quase 0,03 segundos na instância com 5.000 itens. Essa tendência mostra que, apesar de eficiente para problemas menores, o algoritmo guloso pode levar a um custo computacional mais alto em instâncias maiores, reforçando a importância da análise comparativa com outras estratégias.

INSTÂNCIA	MÉDIA TESTES 2 A 6
10	0,00000158s
50	0,00001198s
100	0,00002822s
200	0,00011810s
300	0,00026186s
500	0,00060122s
750	0,00150112s
1000	0,00142232s
1250	0,00393658s
1500	0,00125784s
2000	0,00794064s
2500	0,00493984s
3000	0,00561677s
4000	0,02567756s
5000	0,02993082s

Quadro 3. Média dos testes 2 ao 6 utilizando estratégia gulosa

O Quadro 4 apresenta os resultados dos Testes 1 a 3 utilizando a estratégia transformar para conquistar. Assim como nos quadros anteriores, o Teste 1 foi descartado por ser a primeira execução do programa, que pode sofrer pequenas interferências do sistema. Os resultados mostram que os tempos de execução com essa abordagem são extremamente baixos em todas as instâncias testadas, mesmo nas maiores. Na maioria dos casos, os valores ficaram abaixo de 0,00001 segundos.

Embora existam pequenas variações entre os testes, principalmente em instâncias como 1250 e 2000, elas são mínimas e não comprometem a estabilidade geral da estratégia. Isso reforça a eficiência do uso da ordenação inicial com *Quick Sort*, que, apesar de adicionar um custo teórico de O(n log n), na prática apresenta tempos muito menores em comparação com a estratégia gulosa sem ordenação. Esse quadro evidencia o bom desempenho do algoritmo mesmo em instâncias grandes, indicando que a estratégia é bastante eficiente e escalável.

INSTÂNCIA	TESTE 1 (DESCARTADO)	TESTE 2	TESTE 3
10	0,00000020s	0,00000020s	0,00000010s
50	0,00000030s	0,00000060s	0,00000040s
100	0,00000040s	0,00000040s	0,00000040s
200	0,00000090s	0,00000090s	0,00000090s
300	0,00000110s	0,00000110s	0,00000120s
500	0,00000160s	0,00000160s	0,00000160s
750	0,00000270s	0,00000260s	0,00000260s
1000	0,00000150s	0,00000150s	0,00000150s
1250	0,00000410s	0,00000800s	0,00000400s
1500	0,00000120s	0,00000120s	0,00000120s
2000	0,00000650s	0,00000960s	0,00000650s
2500	0,00000230s	0,00000230s	0,00000220s
3000	0,00000210s	0,00000200s	0,00000210s
4000	0,00001260s	0,00000740s	0,00000740s
5000	0,00000640s	0,00000640s	0,00001020s

Quadro 4. Testes 1 ao 3 utilizando transformar para conquistar

O Quadro 5 apresenta os resultados dos Testes 4 ao 6 para a estratégia transformar para conquistar (modo 2). Assim como observado anteriormente, os tempos de execução continuam extremamente baixos em todas as instâncias, o que reforça o excelente desempenho dessa abordagem. Em praticamente todos os casos, os tempos permanecem na casa dos microssegundos, mesmo quando o número de itens ultrapassa 1000.

As pequenas variações entre os testes, como em 4000 itens (de 0,00000740s a 0,00001190s), são esperadas e podem ocorrer por fatores externos, como pequenas oscilações no uso de recursos do sistema. Apesar disso, essas diferenças são pequenas o suficiente para não afetar a interpretação geral de que o algoritmo é estável e muito eficiente em termos de tempo de execução.

O padrão de crescimento dos tempos permanece suave, sem saltos bruscos, o que demonstra que a estratégia transformar para conquistar oferece desempenho consistente, mesmo à medida que o tamanho do problema aumenta.

INSTÂNCIA	TESTE 4	TESTE 5	TESTE 6
10	0,00000020s	0,00000020s	0,00000020s
50	0,00000030s	0,00000030s	0,00000040s
100	0,00000060s	0,00000030s	0,00000040s
200	0,00000080s	0,00000090s	0,00000090s
300	0,00000160s	0,00000120s	0,00000110s
500	0,00000160s	0,00000160s	0,00000160s
750	0,00000260s	0,00000270s	0,00000260s
1000	0,00000150s	0,00000150s	0,00000150s
1250	0,00000410s	0,00000420s	0,00000418s
1500	0,00000110s	0,00000120s	0,00000110s
2000	0,00000650s	0,00000660s	0,00000660s
2500	0,00000228s	0,00000220s	0,00000220s
3000	0,00000210s	0,00000210s	0,00000210s
4000	0,00000740s	0,00001190s	0,00000750s
5000	0,00000640s	0,00000630s	0,00000640s

Quadro 5. Testes 4 ao 6 utilizando transformar para conquistar

O Quadro 6 apresenta a média dos tempos de execução dos Testes 2 ao 6 utilizando a estratégia transformar para conquistar. Observa-se que os tempos médios permanecem extremamente baixos, mesmo à medida que o número de itens aumenta nas instâncias.

A média de tempo para a menor instância (10 itens) foi de apenas 0,00000018 segundos, enquanto para a maior instância (5000 itens), o tempo médio foi de 0,00000714 segundos. Esses valores demonstram que, apesar do aumento significativo no número de elementos, o tempo de execução cresce de forma bastante controlada e suave.

Além disso, não foram observadas oscilações bruscas ou fora do padrão, o que reforça a estabilidade e previsibilidade do desempenho dessa abordagem. Isso mostra que a combinação entre a ordenação com *Quick Sort* e a seleção sequencial de itens funciona de forma eficiente mesmo em escalas maiores.

Esses dados confirmam que a estratégia transformar para conquistar é muito eficiente em termos de tempo e oferece um desempenho consistentemente superior em relação à estratégia gulosa sem ordenação.

INSTÂNCIA	MÉDIA TESTES 2 A 6
10	0,00000018s
50	0,00000040s
100	0,00000042s
200	0,00000088s
300	0,00000124s
500	0,00000160s
750	0,00000262s
1000	0,00000150s
1250	0,00000490s
1500	0,00000116s
2000	0,00000716s
2500	0,00000224s
3000	0,00000208s
4000	0,00000832s
5000	0,00000714s

Quadro 6. Média dos testes 2 ao 6 utilizando transformar para conquistar

A Tabela 1 apresenta uma comparação direta entre os tempos médios de execução obtidos pela estratégia gulosa e pela abordagem transformar para conquistar (TC) nas instâncias testadas. A diferença entre os tempos está expressa em segundos, evidenciando a economia de tempo alcançada por cada abordagem.

Em todas as 15 instâncias avaliadas, a estratégia TC apresentou desempenho superior, com tempos médios de execução significativamente menores do que a estratégia gulosa. Mesmo nas menores instâncias, como a de 10 itens, a diferença de tempo já é perceptível (0,00000140s). À medida que o número de itens aumenta, essa diferença se torna ainda mais expressiva, como ocorre nas instâncias de 3000 (diferença de 0,00561469s) e 5000 itens (diferença de 0,02992368s).

Essa consistência em favor da estratégia TC indica não apenas maior eficiência no tempo de execução, mas também um comportamento mais escalável e estável, mesmo em instâncias de grande porte. A ordenação inicial dos itens com *Quick Sort*, apesar de representar um custo teórico adicional, acaba compensando ao reduzir drasticamente o tempo necessário para as etapas subsequentes do algoritmo.

O fato de a coluna “Mais Rápido” apontar a estratégia TC como vencedora em todas as linhas reforça a conclusão de que, em termos de tempo, essa abordagem é mais vantajosa para resolver o problema da mochila binária nas condições testadas.

Tabela 1. Comparação das médias da Estratégia Gulosa e Transformar para Conquistar

Instância	Média Estratégia Gulosa	Média TC	Diferença	Mais Rápido
10	0,00000158s	0,00000018s	0,00000140s	TC
50	0,00001198s	0,00000040s	0,00001158s	TC
100	0,00002822s	0,00000042s	0,00002780s	TC
200	0,00011810s	0,00000088s	0,00011722s	TC
300	0,00026186s	0,00000124s	0,00026062s	TC
500	0,00060122s	0,00000160s	0,00059962s	TC
750	0,00150112s	0,00000262s	0,00149850s	TC
1000	0,00142232s	0,00000150s	0,00142082s	TC
1250	0,00393658s	0,00000490s	0,00393168s	TC
1500	0,00125784s	0,00000116s	0,00125668s	TC
2000	0,00794064s	0,00000716s	0,00793348s	TC
2500	0,00493984s	0,00000224s	0,00493760s	TC
3000	0,00561677s	0,00000208s	0,00561469s	TC
4000	0,02567756s	0,00000832s	0,02566924s	TC
5000	0,02993082s	0,00000714s	0,02992368s	TC

A Figura 1 mostra a relação entre o tamanho das instâncias do problema da mochila binária e o tempo médio de execução das duas estratégias avaliadas: Gulosa e Transformar para Conquistar (TC). Observa-se que a linha azul, correspondente à estratégia gulosa, apresenta crescimento acentuado à medida que o número de itens aumenta, o que é compatível com sua complexidade teórica $O(n^2)$. Já a linha vermelha, representando a estratégia TC, mantém-se praticamente constante e próxima de zero em todas as instâncias, confirmando seu comportamento mais eficiente de $O(n \log n)$.

Apesar da tendência geral de crescimento, a estratégia gulosa apresenta alguns pontos com redução no tempo médio, mesmo em instâncias maiores. Por exemplo, o tempo da instância de 1500 itens foi menor que o da instância de 1250; o mesmo ocorreu entre 2000, 2500 e 3000 itens. Esses comportamentos são atribuídos à natureza dos dados de entrada. Como a estratégia gulosa depende diretamente da razão valor/peso dos itens, é possível que certos conjuntos de dados tenham exigido menos comparações ou resultaram em menos iterações completas, mesmo com mais itens.

Além disso, fatores como a localização dos itens mais vantajosos, o número de descartes e a distribuição de pesos podem afetar o fluxo de execução, causando variações nos tempos. Como o algoritmo guloso não realiza ordenação, ele repete buscas sequenciais, que podem ser mais ou menos custosas conforme a ordem e o conteúdo dos dados.

Já a estratégia TC não apresenta esse tipo de variação acentuada, pois sua lógica baseada em pré-ordenação garante maior previsibilidade e desempenho constante, mesmo com diferentes entradas.

Esse comportamento reforça a importância de se considerar não apenas a complexidade assintótica, mas também a natureza dos dados e o estilo de implementação ao avaliar o desempenho prático de algoritmos.

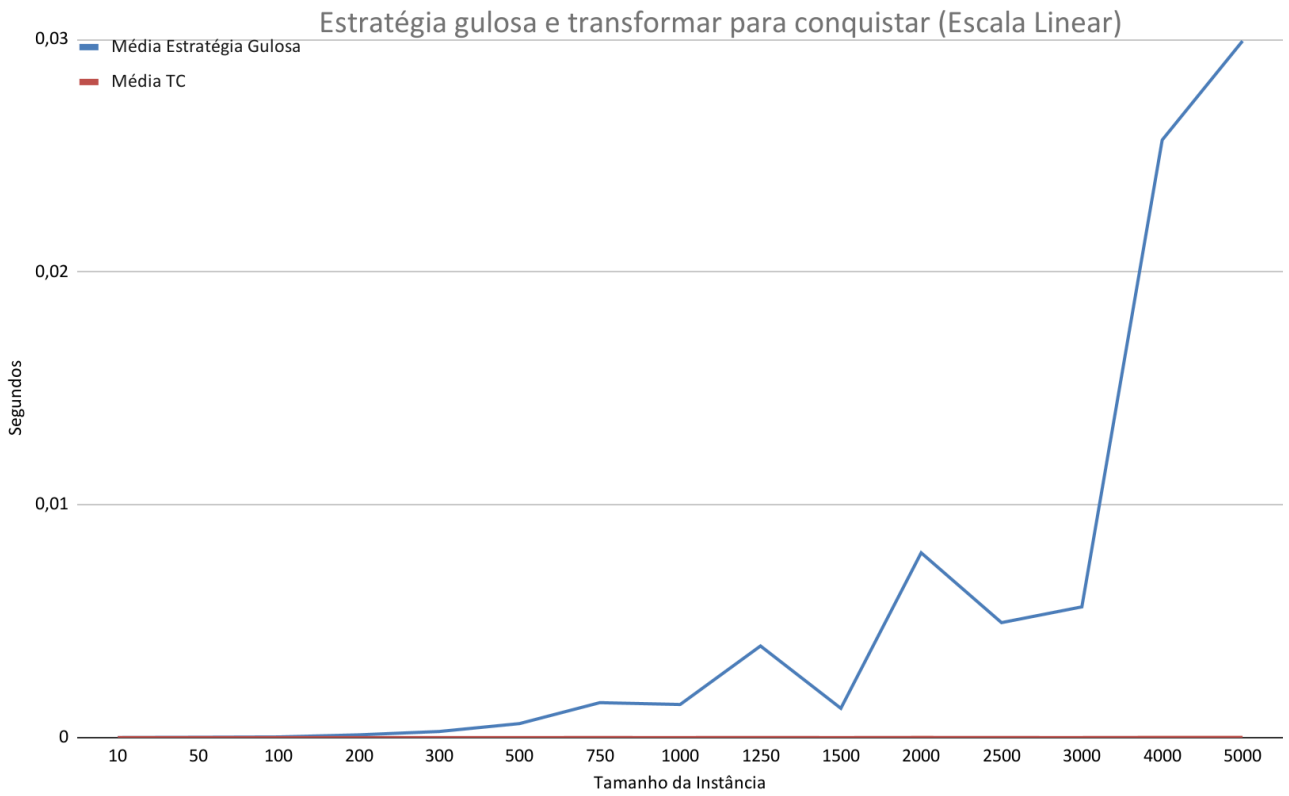

Figura 1. Comparação do tempo médio de execução em função do tamanho da instância (estratégia gulosa x transformar para conquistar)

A Figura 2 apresenta os mesmos dados da Figura 1, porém com o eixo Y em escala logarítmica, o que permite visualizar com maior clareza as diferenças entre as duas abordagens, especialmente para a estratégia Transformar para Conquistar (TC), cujos valores são extremamente baixos e na escala linear apareciam próximos ao eixo. Observa-se que a estratégia gulosa mantém uma tendência geral de crescimento, condizente com sua complexidade quadrática $O(n^2)$, porém apresenta quedas pontuais, como entre as instâncias de 1250 e 1500 itens, quando o tempo médio reduz de 0,00393658s para 0,00125784s, e entre 2000 e 2500 itens, com redução de 0,00794064s para 0,00493984s. Esses comportamentos ocorrem porque o custo efetivo do algoritmo não depende apenas do número de itens, mas também da disposição interna dos dados, que influencia o número de operações realizadas.

Já a estratégia TC apresenta tempos extremamente baixos, embora com variações discretas ao longo das instâncias. Nota-se aumento entre 100 e 200 itens, seguido por um acréscimo de 500 para 750 itens e uma queda inesperada em 1000 itens. Em 1250 itens, ocorre um aumento expressivo em relação ao anterior, enquanto 1500 itens apresenta forte redução. Na sequência, observa-se um pico em 2000 itens, queda em 2500 e 3000 itens, novo aumento em 4000 itens e redução final em 5000 itens. Essas oscilações decorrem de fatores como a escolha do pivô e a distribuição dos dados durante a ordenação pelo *Quick Sort*, além de possíveis variações no ambiente de

execução. Apesar disso, os tempos permanecem na ordem de microssegundos, confirmando a eficiência da estratégia TC.

A utilização da escala logarítmica evidencia a discrepância entre as duas abordagens, revelando que, enquanto a estratégia gulosa cresce várias ordens de magnitude, a TC se mantém praticamente estável, reforçando sua escalabilidade superior.

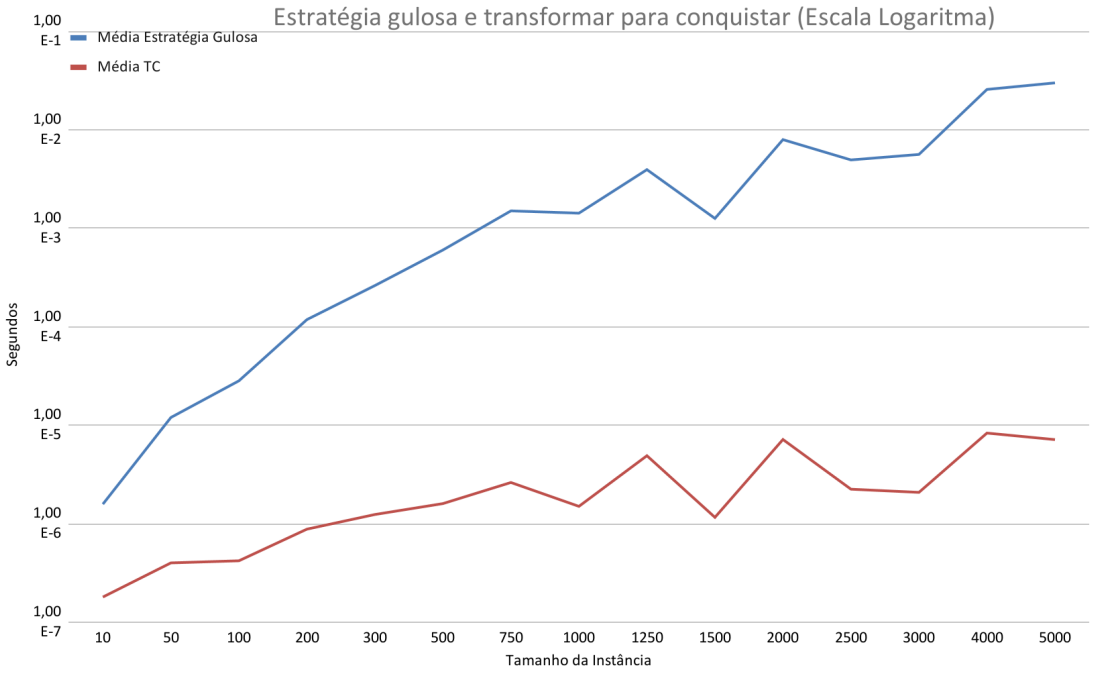

Figura 2. Comparação do tempo médio de execução em escala logarítmica (estratégia gulosa x transformar para conquistar)

A Figura 3 apresenta a comparação entre os tempos médios de execução das estratégias Gulosa e Transformar para Conquistar (TC), utilizando escala linear e intervalos fixos de 1000 unidades no eixo X, que representa o tamanho da instância do problema da mochila binária. Este formato foi adotado para permitir uma visualização mais regular do comportamento das estratégias em diferentes escalas de entrada.

Observa-se que a linha azul, correspondente à estratégia Gulosa, apresenta uma tendência geral de crescimento com o aumento do número de itens, compatível com sua complexidade assintótica $O(n^2)$. No entanto, o gráfico também evidencia oscilações, com quedas pontuais no tempo médio entre algumas instâncias consecutivas, como entre 1250 e 1500, e entre 2000 e 2500 itens. Tais variações decorrem do fato de que o desempenho do algoritmo guloso depende fortemente da disposição interna dos dados,

já que a seleção sequencial sem ordenação pode resultar em menos iterações completas em certos casos.

Por outro lado, a linha vermelha, que representa a estratégia Transformar para Conquistar, mantém-se praticamente sobre o eixo X ao longo de toda a faixa de instâncias. Isso demonstra que os tempos de execução dessa abordagem permanecem extremamente baixos, mesmo em instâncias de grande porte, refletindo sua complexidade empírica O(n log n) e sua alta eficiência prática. A presença de ordenação prévia com Quick Sort contribui para essa estabilidade, reduzindo o impacto da variação dos dados de entrada.

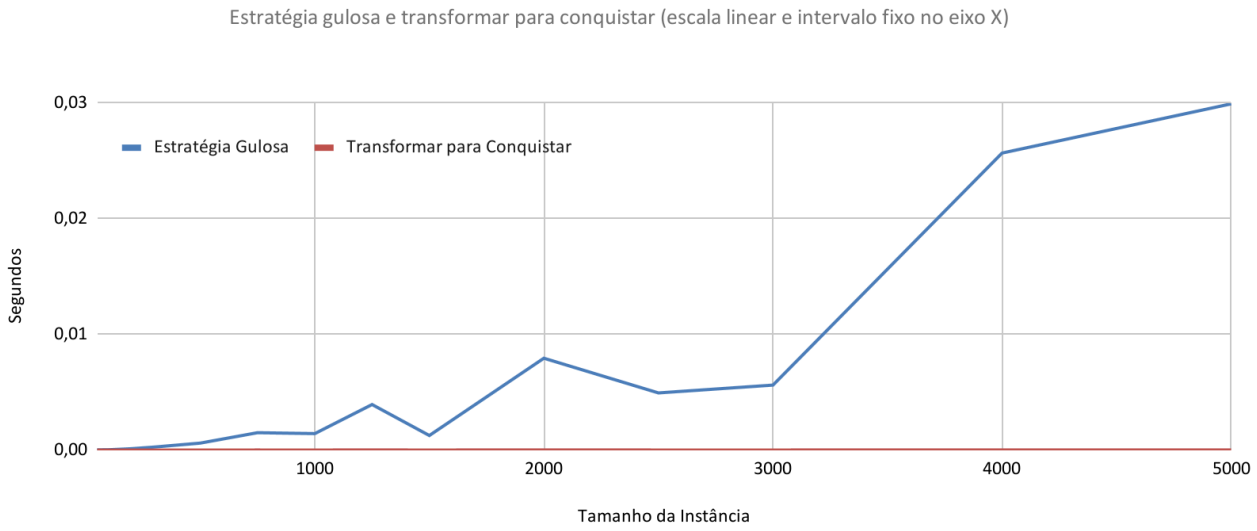

Figura 3. Comparação do tempo médio de execução das estratégias gulosa e transformar para conquistar (escala linear e intervalos fixos no eixo X)

A Figura 4 apresenta a comparação entre as estratégias Gulosa e Transformar para Conquistar (TC) com o eixo Y em escala logarítmica e intervalos fixos de 1000 unidades no eixo X. Essa representação permite uma visualização mais clara das ordens de magnitude dos tempos de execução, sobretudo nos menores valores, que tendem a se comprimir em escalas lineares.

A estratégia Gulosa, representada pela linha azul, continua exibindo crescimento expressivo conforme o número de itens aumenta, evidenciando sua complexidade quadrática $O(n^2)$. Mesmo com variações entre as instâncias, a tendência geral é de elevação progressiva dos tempos. Já a linha vermelha, correspondente à estratégia TC, permanece em uma faixa muito inferior, com tempos consistentemente baixos, reforçando seu bom desempenho prático mesmo com a inserção da etapa de ordenação (Quick Sort), cuja complexidade é O(n log n).

A escala logarítmica torna mais visível a separação entre os desempenhos das duas abordagens. Enquanto a Gulosa cresce rapidamente em várias ordens de grandeza,

a TC apresenta uma curva praticamente plana, com pequenas oscilações esperadas devido à influência de fatores como variação nos dados de entrada e comportamento do ambiente de execução. Essa visualização confirma a maior escalabilidade da estratégia TC e sua superioridade prática na resolução do problema da mochila binária.

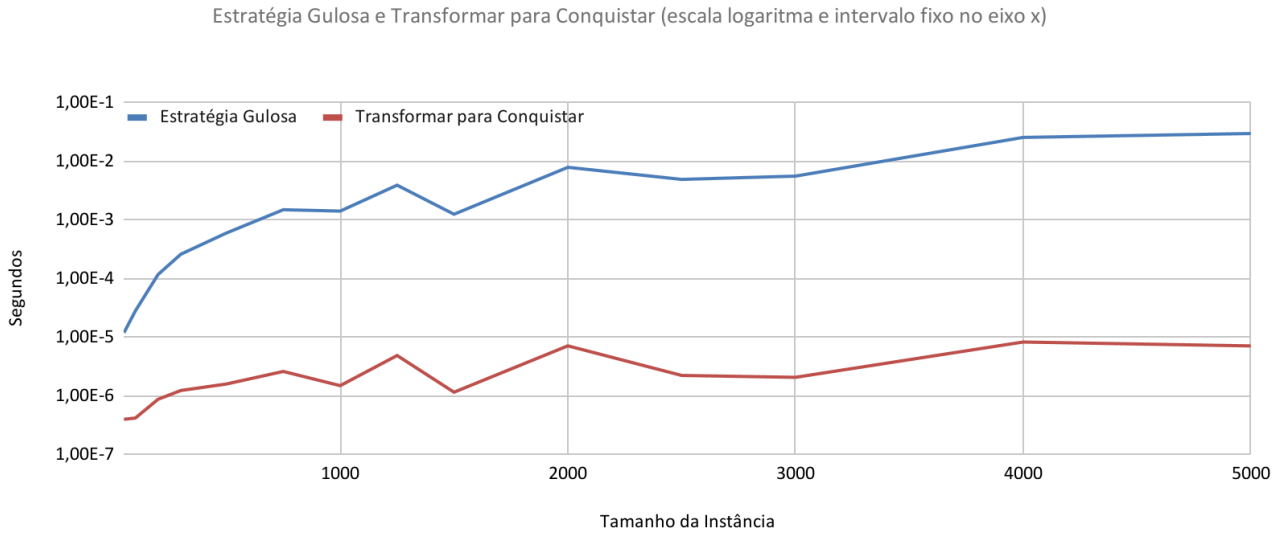

Figura 4. Comparação do tempo médio de execução das estratégias gulosa e transformar para conquistar (escala logarítmica e intervalos fixos no eixo X)

5. Conclusão

A análise comparativa entre as estratégias gulosa e transformar para conquistar na resolução do problema da mochila binária permitiu observar diferenças em termos de desempenho prático e complexidade algorítmica. A estratégia gulosa, apesar de sua simplicidade e facilidade de implementação, demonstrou crescimento acentuado no tempo de execução à medida que o número de itens aumentou, comportamento compatível com sua complexidade assintótica $O(n^2)$. Embora eficiente em instâncias pequenas, seu desempenho se torna cada vez mais limitado em problemas de maior escala, sendo também mais sensível à disposição dos dados de entrada, o que ocasionou oscilações inesperadas nos tempos médios.

Em contraste, a estratégia transformar para conquistar, que incorpora a ordenação inicial dos itens por custo-benefício com o algoritmo Quick Sort, mostrou-se amplamente superior. Com complexidade $O(n \log n)$, essa abordagem manteve tempos extremamente baixos e estáveis, mesmo nas maiores instâncias testadas. Embora tenha apresentado variações pontuais nos tempos de execução, atribuídas a fatores como a distribuição dos dados e a aleatoriedade na escolha do pivô, essas flutuações não

comprometeram sua eficiência geral, permanecendo sempre em uma ordem de magnitude significativamente inferior à da abordagem gulosa.

Os resultados obtidos evidenciam que, para o problema da mochila binária, a ordenação prévia dos dados contribui de maneira decisiva para a redução do tempo de execução, mesmo adicionando um custo computacional teórico. Dessa forma, conclui-se que a estratégia transformar para conquistar oferece uma solução mais escalável, robusta e eficiente, sendo mais adequada para aplicações práticas que demandam rapidez e estabilidade, especialmente em contextos com grandes volumes de dados. Além disso, a análise empírica demonstrou a importância de avaliar o desempenho dos algoritmos não apenas com base em sua complexidade teórica, mas também levando em consideração a implementação, o perfil das instâncias e os recursos computacionais disponíveis.

Referências

Abidin, S. (2017). Greedy approach for optimizing 0-1 knapsack problem. Communications on Applied Electronics (CAE), 7(6):1–3.

Brun, A. L. (2025). Transformar para conquistar. Apresentação de slides. Aula 14 - Projeto e Análise de Algoritmos.

Durmuş, B., Isci Guneri, O., and İncekirik, A. (2019). Comparison of classic and greedy heuristic algorithm results in integer programming: Knapsack problems. Mugla Journal of Science and Technology, 5.

Folador, J. P., Pena Neto, L. N., and Jorge, D. C. (2014). Aplicativo para análise comparativa do comportamento de algoritmos de ordenação. Revista Brasileira de Computação Aplicada, 6(2):76–86.

Pradhan, T., Israni, A., and Sharma, M. (2014). Solving the 0–1 knapsack problem using genetic algorithm and rough set theory. pages 1120–1125.

Wang, Y. (2023). Review on greedy algorithm. Theoretical and Natural Science, 14:233–239.